

A New Application for Displaying and Fusing Multimodal Data Sets

Karl G. Baum^{*ac}, María Helguera^a, Andrzej Krol^b

^aRochester Institute of Technology, 54 Lomb Memorial Drive, Rochester, NY, USA 14623-5604;

^bSUNY Upstate Medical University, 750 East Adams Street, Syracuse, NY USA 13210-2306;

^cKGB Technologies, 421 Countess Drive, West Henrietta, NY USA 14586-9423

ABSTRACT

A recently developed, freely available, application specifically designed for the visualization of multimodal data sets is presented. The application allows multiple 3D data sets such as CT (x-ray computer tomography), MRI (magnetic resonance imaging), PET (positron emission tomography), and SPECT (single photon emission tomography) of the same subject to be viewed simultaneously. This is done by maintaining synchronization of the spatial location viewed within all modalities, and by providing fused views of the data where multiple data sets are displayed as a single volume.

Different options for the fused views are provided by plug-ins. Plug-ins typically used include color-overlays and interlacing, but more complex plug-ins such as those based on different color spaces, and component analysis techniques are also supported.

Corrections for resolution differences and user preference of contrast and brightness are made. Pre-defined and custom color tables can be used to enhance the viewing experience. In addition to these essential capabilities, multiple options are provided for mapping 16-bit data sets onto an 8-bit display, including windowing, automatically and dynamically defined tone transfer functions, and histogram based techniques.

The 3D data sets can be viewed not only as a stack of images, but also as the preferred three orthogonal cross sections through the volume. More advanced volumetric displays of both individual data sets and fused views are also provided. This includes the common MIP (maximum intensity projection) both with and without depth correction for both individual data sets and multimodal data sets created using a fusion plug-in.

Keywords: multimodal, visualization, fusion, software, biomedical, dynamic range, display, medical imaging

1. INTRODUCTION

It has now become clear to the medical community that in certain situations a multiple image approach has significant advantages over single image approaches. Often images of the same patient are acquired using the same instrument at different times, allowing the change that occurs between the times of acquisition to be observed. It is also common for multiple images to be acquired with a single instrument, but with different acquisition parameters or protocols. For example, in magnetic resonance imaging, acquiring multiple images using different pulse sequences allows tissues to be visualized with different contrasts, which may aid in differentiating the tissues. Within the last decade the advantages of acquiring images from multiple instruments have begun to outweigh the costs. Images from different modalities, such as x-ray computed tomography (CT), magnetic resonance imaging (MRI), single photon emission computer tomography (SPECT), positron emission tomography (PET), and ultrasound are used to acquire complimentary information. For example an MRI or CT image might be acquired to obtain the anatomical information followed by the acquisition of one or more PET or SPECT images to obtain information on the physiological behavior in the areas of interest.

There has been a significant amount of effort spent registering images from different modalities. Registration is the process of obtaining the spatial relationships between images, and manipulating the images so that the corresponding pixels in them represent the same physical locations. There has been significantly less effort however on finding the best way to utilize the registered information. This includes efforts on image fusion, which is the process of taking multiple images and combining them into a single image, and work on interfaces for analyzing multiple medical images simultaneously.

* kgb5056@rit.edu; www.kgbtechnologies.com/fusionviewer

To the author's knowledge there is not a readily available tool, for use by the medical research community, which is designed for studying multimodal or multi-image data sets. The application presented here is designed to address this need. It is designed to allow easy side-by-side evaluation of multiple data sets, and provides tools for studying fusion techniques and the volumetric display of fused data sets.

The remainder of this article focuses on the features of the Fusion Viewer software package. The design of the application will be briefly discussed before focusing on its capabilities. Working with volumetric data sets and different tools for viewing them will be covered. Options for controlling the viewing experience including spatial scaling, brightness and contrast settings, dynamic range options, and coloring will be discussed. After a complete coverage of the tools for viewing a single volume, tools for viewing multiple volumes will be discussed. This will include the synchronizing of data sets, and tools for fusion. Implementation of fusion plug-ins will be covered in detail with code examples. The paper will conclude with an overview of the software and information on obtaining it.

2. APPLICATION DESIGN

The application, Fusion Viewer, was designed and implemented with a modular object oriented design. It was decided that the application should be implemented on the .NET platform, in its language of choice, C#. This insures that the application can be widely used due to the unique capability of a .NET application to be both operating system, and machine architecture independent. The growing popularity of the .NET platform ensures that the application will be supported for the foreseeable future.

Novel fusion techniques, both of the raw image data and of projection data, can be quickly and easily implemented as plug-ins. The plug-ins simply need to be placed in the program directory and no recompilation of the application source code is necessary. Due to the language independent nature of the .NET platform implementers of plug-ins can work in the programming language of their choice [1].

Fusion Viewer is designed to process data on an as needed basis. By only processing the portions of the image volume that are currently visible to the user the images can be resized, converted to 8-bit, undergo contrast and brightness adjustments, have color tables applied and get fused with other images in real time. By processing on an as needed basis a highly interactive environment is available to the user.

Difficult installation of research tools often limits their acceptance by the medical community. The .NET platform makes the distribution of the Fusion Viewer software trivial. The application only needs to be compiled (to MSIL) once, after which it can be run on any computer with the .NET framework installed. This means it can simply be downloaded and run, the user does not need to perform a complicated compilation process.

3. WORKING WITH VOLUMETRIC DATA SETS

3.1 Importing and Exporting Data

The 'File' menu of Fusion Viewer provides access to the import and export features of the application. While Fusion Viewer supports two-dimensional images, it is designed specifically for three-dimensional data sets. The read and write capabilities of the software are provided by the FreeImage library [2, 3]. FreeImage is a cross-platform library designed to allow popular graphics image formats to be easily used in other applications. This means that with little effort the Fusion Viewer application can be modified to read and write any of the more than 20 image formats supported by FreeImage.

Currently only the capability for importing multi-page tiff files is included in the release version of the software, but work on supporting other formats such as DICOM is underway. When exporting images from Fusion Viewer you have the choice of exporting the entire volume as you have manipulated it for display, or just the currently visible portion of the volume (ex. a slice from a stack view, or the current projection from a projection view).

3.2 Displaying a Volumetric Data Set

By default when a volume is first opened it is displayed using a standard three orthogonal cross section view. Figure 1 is a screenshot that shows volumes being displayed using this method. Each displayed volume is shown as three images, each representing a cross-section through the volume, axial, sagittal, and coronal. The three images shown are selected

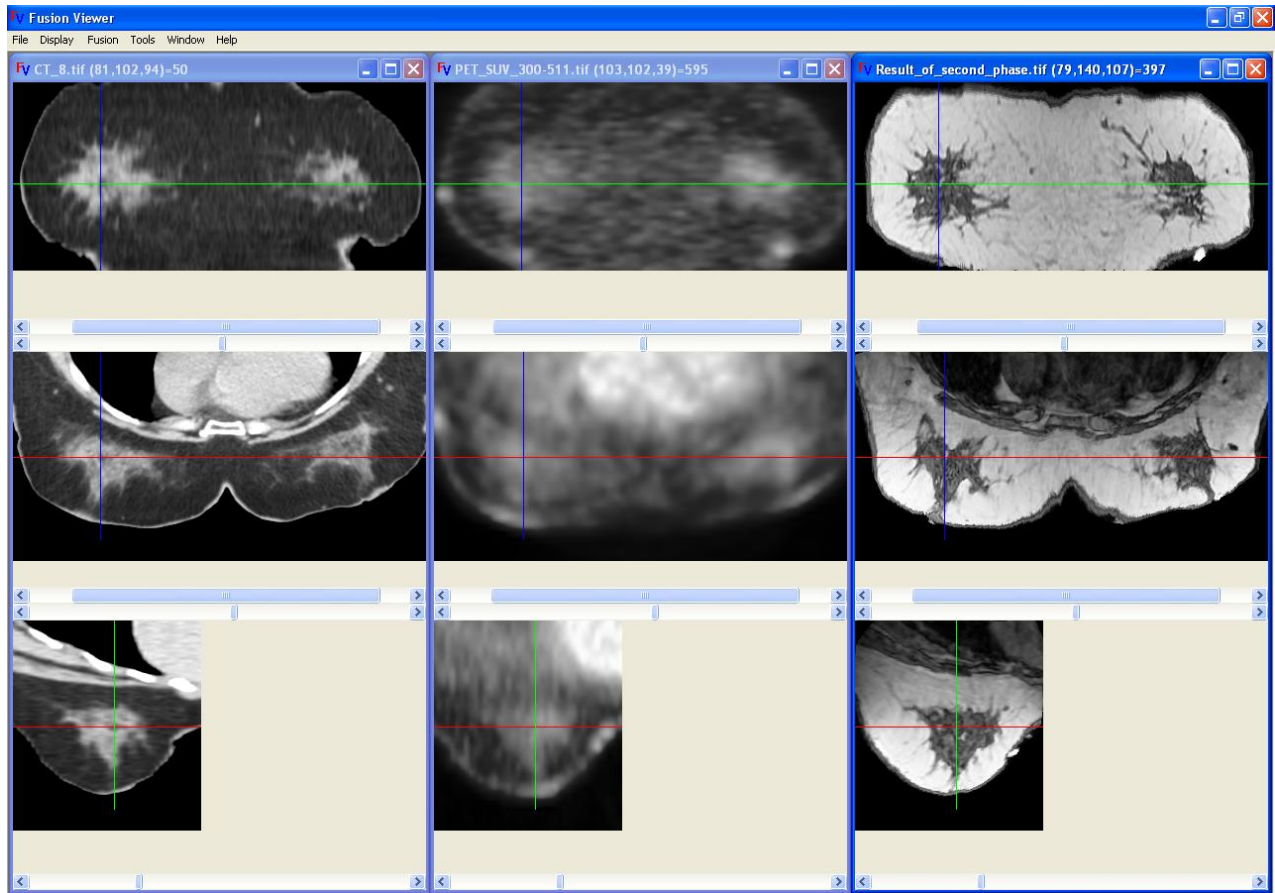


Fig. 1. Volumes displayed using the three orthogonal cross section view. The top images are coronal slices, the middle are axial slices, and the bottom are sagittal slices. From left to right a CT data set, a PET data set, and a MRI data set are shown.

in such a way that they pass orthogonally through a focus point in the volume. The cross-hairs shown on each image indicate the point of focus, and the two lines of each cross-hair represent the locations of the other two displayed slices.

The focus can be changed by either manually entering a coordinate or by clicking on one of the three orthogonal views. Clicking on a view will move the cross-hair to the point clicked, and will change the other two views so that they represent the slices at the new location of the cross-hair. The cross-hairs can be made invisible so that they are not a distraction when reading the images. Lastly, a scroll bar located under each of the views allows the location of the slice shown along the associated axis to be changed. For example, scrolling the scroll bar under the coronal view will result in the neighboring coronal slices being shown.

A three orthogonal cross section view allows a volumetric data set to be easily navigated while being presented in full detail. Every voxel can be independently examined and easily located. This is not always the case with other volumetric views such as projections or surface renderings.

Other views can be found on the 'Display' menu. A standard stack view is available, where the volumetric data is presented as a stack of images with different locations along the coronal, the sagittal, or the axial axis. The only other way to view a volume data set in the current version of Fusion Viewer is by using projections. When using projections a new volume is created which consists of projections of the current volume acquired from different directions. These projections are created using standard ray tracing techniques [4]. Successively showing different projections give the appearance of a rotating three-dimensional volume. Projection techniques can be used to create what looks like a rotating three dimensional volume on a two dimensional display.

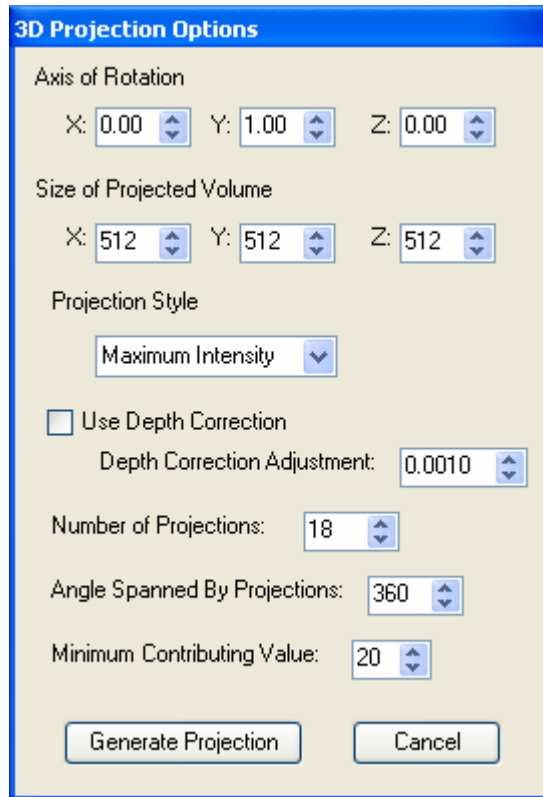


Fig. 2. Dialog that allows the user to change the volume projection options.

The interface for setting the projection options is shown in Figure 2. The first option is the axis with which to have the projections revolve around. A series of projections will be taken at different angles around this axis. Unlike most projection programs, Fusion Viewer can create the projections along any arbitrary axis. The second option is the size of the projected volume. Typically this is set to a size greater than the extent of the volume when rotated around the selected axis.

The projection style allows the property that we are projecting to be selected. Currently either the maximum intensity or the mean intensity can be projected. The maximum intensity projection (MIPS) is the most common tool used to generate a three-dimensional looking display of medical data sets [5]. The mean intensity projection is an alternative for denser imaging modalities, such as MRI, that give poor results when using a maximum intensity projection. The algorithms were implemented in such a way that very little new code needs to be written in order to implement other projection styles.

The depth correction option can be used to increase the three-dimensional feel of the projections. When using depth correction voxels closer to the source of the traced rays are given more weight than voxels further from the ray source. In other words the voxels at the front of the volume when viewed at the current angle are weighted more than the voxels at the back of the volume. The difference in weight is determined by the depth correction adjustment option.

Other options include the number of projections to take, and the total angle to be covered by the projections. The last parameter, the minimum contributing value, can be used to ignore voxels with low signal. This can often be used to prevent noise in the background from contributing to the projections.

3.3 Image Adjustments for Improved Display

Most medical images, as acquired, are not ready to be examined using the previously discussed display options. A few simple steps can be taken to prepare the data for analysis. All of the options discussed here are available from the 'Display' menu of the Fusion Viewer program. The first issue that needs addressed is the image resolution. Typically

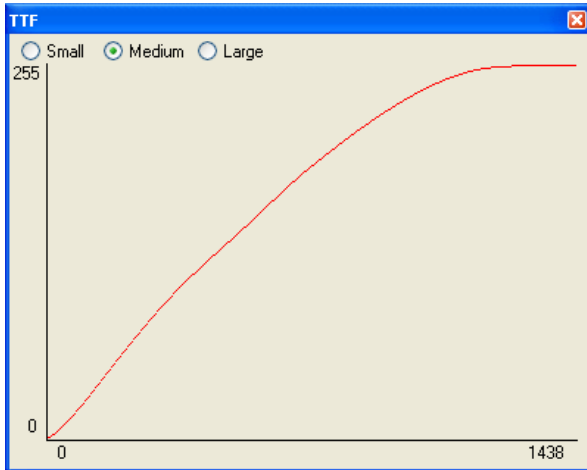


Fig. 3. The tone transfer function used to map the original 1439 intensities in the MRI data set shown in Figure 1 to the 256 intensities that can be shown by a standard CRT or LCD monitor.

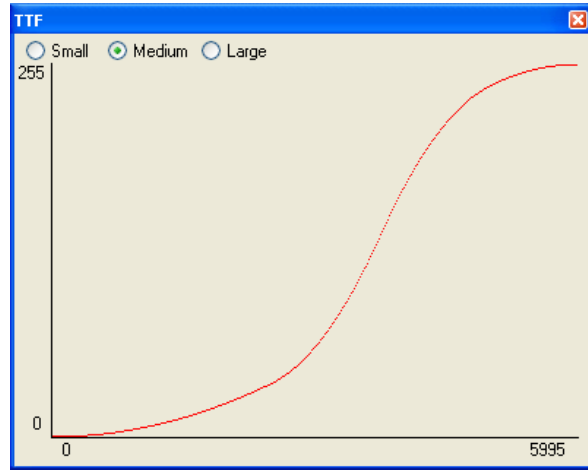


Fig. 4. An example tone transfer function created using the quadratic spline dynamic range technique.

the voxels of a medical data set are not isotropic. Most modalities produce a volume with a higher in-plane resolution than a between plane resolution. If such a volume is treated as though it had isotropic voxels it would appear as if it was being compressed along the axis orthogonal to the acquired planes. To correct this, a scale factor can be set. The scale factor along each axis indicates the amount a voxel needs to be stretched in each direction in order to make it isotropic. Setting the scale factor will allow Fusion Viewer to display the images correctly. The scale factor can also be used for zooming in and out of the image. For example to display a volume one quarter of its size multiply the scale factor of each axis by 0.25, to display the volume twice its size multiply each of the scale factors by 2.

Another important issue that needs to be dealt with is the dynamic range of the volume. Most modalities provide 16-bit data sets. This allows 65536 unique intensities for each voxel. Even though most of the modalities don't use this full range, they do use more than the 256 supported by standard CRT or LCD displays. Finding the best options for mapping 16-bit data sets to an 8-bit display is an active area of research. Fusion Viewer provides the traditional tools to do this as well as some more advanced and novel techniques.

When a global operator is used to map the 16-bit data to 8-bits this mapping can be described by a function known as the tone transfer function (TTF). Figure 3 shows the TTF that was used for displaying the MRI images shown in Figure 1. The values on the horizontal axis correspond to the 1439 intensities found in the original MRI data set, and the values on the vertical axis correspond to the 256 intensities that can be shown on the display. To map a voxel's intensity from the MRI volume to its displayed intensity, move vertically on the TTF plot from the appropriate value. After intersecting the TTF move horizontally until the vertical axis has been crossed. The point at which the vertical axis is crossed represents the intensity that the voxel will be assigned for display.

The traditional linear mapping, of 16-bit data sets to 8-bits, used in medical applications is supported. The range of the intensities in the original data set to be mapped linearly to the display can be either selected by setting a minimum and maximum or a window and level. More advanced histogram based techniques are supported [6] as well as two novel spline based techniques. The histogram techniques are good for maximizing the contrast shown in the displayed images, or displaying images where an appropriate window and level are not known. The spline based techniques are designed to be an alternative to the traditional linear windowing.

For the spline based techniques, the standard window and level settings are used, only instead of linearly mapping the values in the window and setting values outside of the window to 0 and 255, values within the window are mapped to a percentage of the display's intensity range and the values outside of the windows are compressed significantly to the remaining portions of the display's intensity range. An example of a quadratic spline TTF is shown in Figure 4. Here

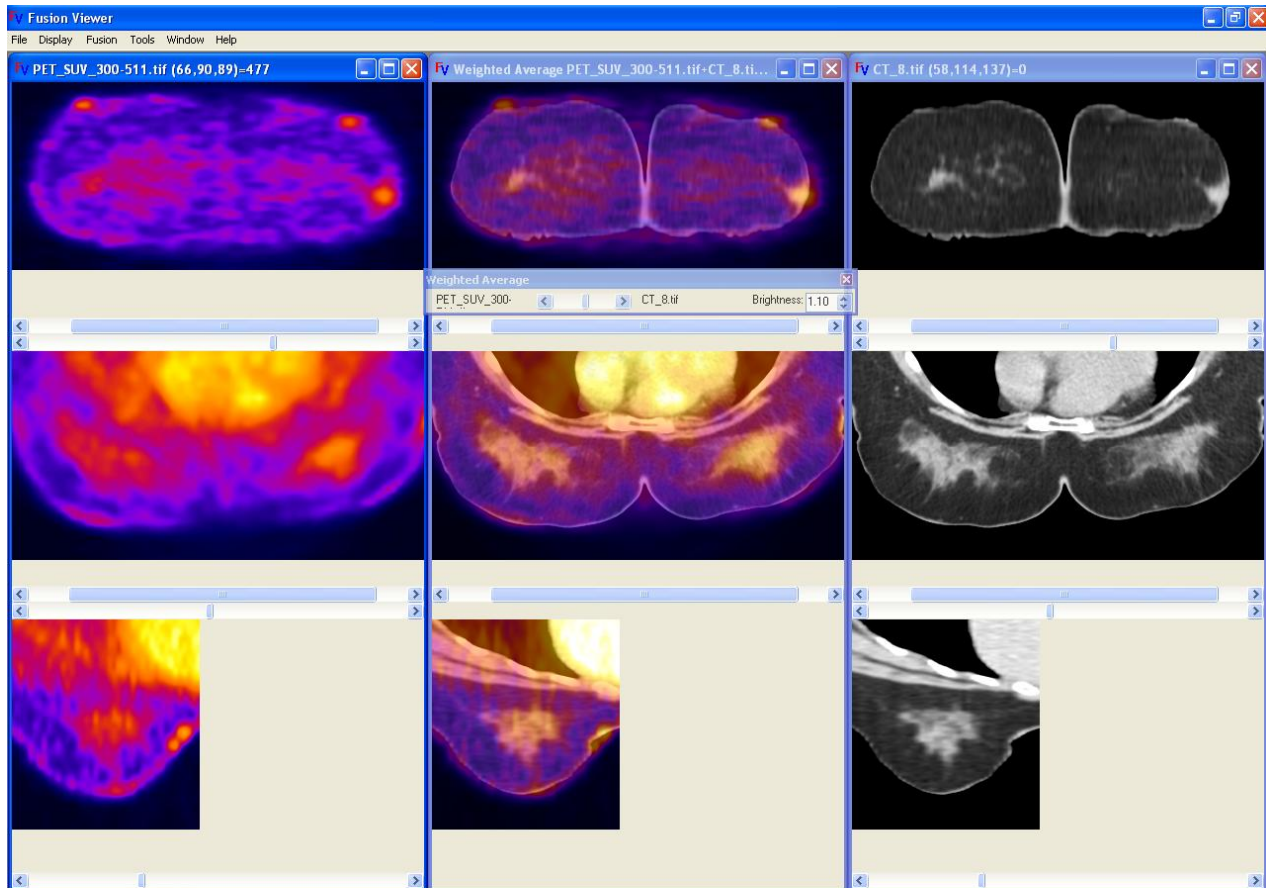
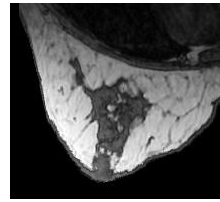


Fig. 5. A screenshot of a fused data set created using the Fusion Viewer software. The three orthogonal images on the left are from the PET data set after the application of the Fire color table, the three orthogonal images on the right are from the CT data set. The three orthogonal images in the center are from fusing the PET and CT images using the weighted average fusion plug-in. In essence the CT image gets colored based on the PET intensity values. Not only can we see the anatomical structure, but we can also see the metabolic activity for each structure.

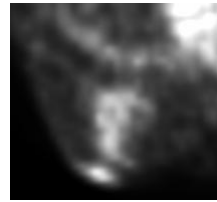
the intensities of interest from 2500 to 5000 are mapped to 80% of the display's intensity range, and the remaining intensities are compressed significantly more and mapped onto the remaining 20% of the display's intensity range. The advantage of the spline based windowing techniques over traditional windowing techniques is their ability to retain a relatively high contrast in the intensity range of interest, while not totally discarding the intensity information that does not fall in this range. Using this technique, what may be important, contextual information is not totally lost. The plug-in interface for Fusion Viewer allows alternative dynamic range techniques, such as the spline based techniques, to be easily implemented.

A few other options for fine tuning the display exist. These include the ability to linearly scale the brightness of the image and the option to set voxels whose intensities exceed the maximum value being displayed to black instead of the usual white. This setting may make reading the images easier on the eye, and takes the natural emphasis off of these voxels that are outside of the range of intensities that are of interest.

The option to present a grayscale image in color is present. Through the use of color tables, also known as look up tables or color maps, the 256 display intensities can be assigned colors. For compatibility Fusion Viewer adopted the color table definition used by the popular ImageJ software package [7]. 22 such color tables are included, but any other one can also be loaded and applied to a grayscale image. The PET data set in Figure 5 is an example of the Fire color table applied to a grayscale image.



Original MRI Image



Original PET Image

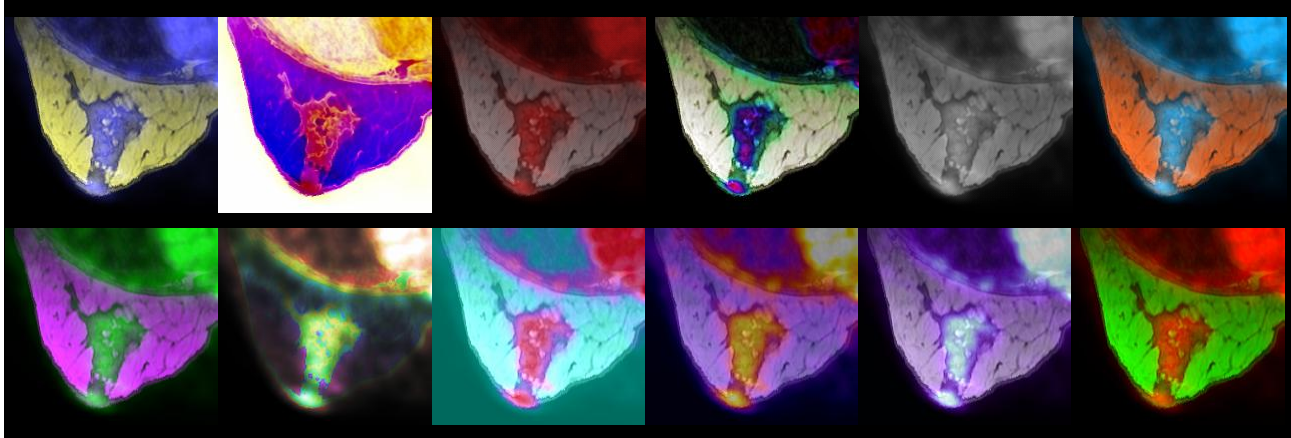


Fig. 6. An assortment of fused images created by fusing the original MRI and PET image shown at the top of the figure. All images were created using fusion plug-ins supplied with Fusion Viewer.

3.4 Viewing Multiple Data Sets

While Fusion Viewer provides a nice tool for examining stand alone medical images, the real strength of the application lies in its ability to simultaneously examine multiple registered data sets. The simplest example of this is when multiple images are open such as in Figure 1. Unless otherwise specified by the user, Fusion Viewer will keep the view of all of the opened volumes synchronized. In other words changing the point of focus for the PET volume will also change the point of focus for the MRI and CT volumes. Similarly, if projections are shown for multiple volumes, changing the angle of projection for one volume will change the angle of the projection shown for all of the other volumes. This allows a registered view of multiple data sets to be maintained as they are examined side by side.

In addition to allowing multiple volumes to be viewed side by side, they can also be combined (fused) and viewed as a single volume. The currently supported fusion techniques are listed under the 'Fusion' menu, but as will be shown later new fusion techniques can be easily implemented using Fusion Viewer's plug-in interface. Upon selecting a fusion technique from the 'Fusion' menu a dialog is displayed. This simple dialog is used to select the volumes to fuse and the role they will play in the fused image. For example when selecting the hue, saturation, lightness plug-in this dialog box allows you to choose which volume should be used for the hue value, which should be used for the saturation value, and which should be used for the value of the lightness.

Figure 5 provides an example of fusing a CT image with an FDG-PET image. Here the weighted average fusion plug-in was used. When using this plug-in the colors of the CT image are averaged with the colors from the PET image using a weighting based on the position of the scrollbar in the window titled 'Weighted Average'. By moving this scrollbar all of the way to the right we see the original CT image, all the way to the left we see the original PET image. Anywhere in between we see a weighted average of the two.

In addition to being able to fuse multiple volumes when they are displayed as three orthogonal cross sections, the fusion plug-ins can be used to fuse projections of the volumes and plug-ins for creating projections of fused volumes exist. Figure 6 shows a subset of the fusion capabilities provided by the current plug-ins. For more information on fusion techniques see [8, 9] and their associated posters which can be found at [10].

```

1. namespace MyPlugin
2. {
3.     public class MyPlugin : FusionViewer.OrthogonalViewFormFused, FusionViewer.IPlugin
4.     {
5.         public int NumberOfSources
6.         {
7.             get { return 2; }
8.         }
9.
10.        public string SourceSelectionMessage
11.        {
12.            get { return "Select 2 source volumes with the same dimension."
13.                + " The first source selected will be used for the red channel,"
14.                + " the second for the blue."; }
15.        }
16.
17.        public System.Type SourceType
18.        {
19.            get { return typeof(FusionViewer.OrthogonalViewFormGreyscale); }
20.        }
21.
22.        public override byte[] RawSliceData(
23.            FusionViewer.SliceDirection sliceDirection, FusionViewer.Point3D focus)
24.        {
25.            byte[] source1 = sources[0].RawSliceData(sliceDirection, focus);
26.            byte[] source2 = sources[1].RawSliceData(sliceDirection, focus);
27.            byte[] buffer = new byte[4 * source1.Length];
28.            for (int i = 0; i < source1.Length; i++)
29.            {
30.                buffer[4 * i] = source2[i];
31.                buffer[4 * i + 1] = 0;
32.                buffer[4 * i + 2] = source1[i];
33.            }
34.            return buffer;
35.        }
36.
37.        protected override void Initialize()
38.        {
39.            this.Title = "My Plugin " + " Red: " + sources[0].Title + " Blue: "
40.                + sources[1].Title;
41.            this.Text = this.Title;
42.        }
43.    }
44. }

```

Fig. 7. The code for a basic fusion plug-in.

3.5 Implementing a New Fusion Plug-in

In this section the required code for a plug-in will be covered. The discussion will pertain to the code shown in Figure 7. This code was written in C#, for a good introduction to C# see [11]. As mentioned before a programmer should be able to write their plug-in in the language of their choice supported by the .NET framework. The sample plug-in discussed here takes two input image volumes and creates a fused color image by assigning one input image to the monitor's red channel, and the other to the monitor's blue channel. Each plug-in will be contained within its own class. References need to exist to System.dll, System.Windows.Forms.dll that are included with the .NET framework, and to FusionViewerResources.dll that is provided as part of the Fusion Viewer software package. When discussing the code, the current line of interest will be identified by its line number. The line numbers are shown in red in Figure 7.

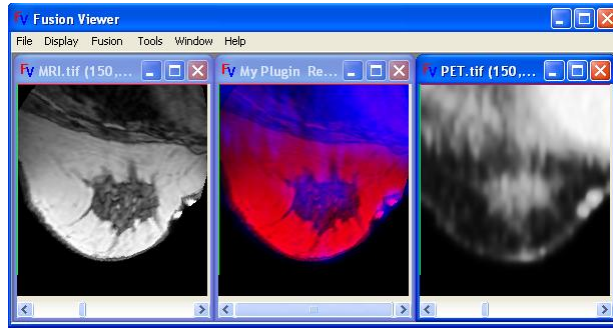


Fig. 8. Screenshot taken of the plug-in defined by the code in Figure 7. The setting to only show the sagittal view was selected from the 'Display' menu.

The code begins with the standard namespace and class definition on line 1 and 3 respectively. The class is required to inherit from `FusionViewer.OrthogonalViewFormFused`, and implement the `FusionViewer.IPlugin` interface. The class needs to have several properties that tell the Fusion Viewer application and the user what the required input images to the plug-in should be. The first property, `NumberOfSources`, gives the number of images the plug-in fuses. The property can be found on line 5. It usually returns a value of two or three, but plug-ins can work with a much larger number of input images as well. For example a plug-in that takes a dynamic PET series and returns a volume representing the peak SUV for each voxel might require a much higher number of input images. The number of input images is only limited by the system memory. The example plug-in requires two input images.

The second required property, located on line 9, returns an instructional message to plug-in users. This property must be called `SourceSelectionMessage` and in our case returns a string telling the user that the first image they select will be used for the red channel and the second for the blue channel.

The last property is `SourceType` located on line 13. This property is used to identify the types of images that the plug-in works with. In this case we specify `FusionViewer.OrthogonalViewFormGreyscale`, which means our plug-in can use any grayscale image as its input. These images can either be a volume loaded into the program or a projection volume created by the program. Other options for the source image types include color images created by other plug-ins or user defined image types.

One of the two functions, `RawSliceData` on line 17, is where the plug-in does its work. This function always receives as input the orientation of the slice currently being viewed, and the focus point. When working with projection volumes the orientation of the slice will be a constant. The plug-in programmer usually does not need to worry about what these parameters represent, since in most cases they will just be passed on.

In this function as well as throughout the rest of the class the plug-in will have access to an array of source images. These are the images that were provided by the user to the plug-in. In lines 19, and 20 we reference the sources images in order to obtain the portions of the images that are currently being viewed by the user. The `RawSliceData` function of each source image provides this data when you pass it the slice orientation and the focus. By simply passing on the `sliceDirection` and `focus` arguments to the `RawSliceData` function for each source the plug-in will have access to the data that needs processed for display. By only processing the data currently being displayed most plug-ins can perform image fusion in real time.

On line 21 we define an array of bytes called `buffer` that will be used to store the fused image data. The `RawSliceData` function of the plug-in will return `buffer` and not have to worry about preparing the data for display or actually displaying it.

In lines 22 through 27 we enter a FOR loop. In the FOR loop a color is assigned to each voxel of the fused image that is currently visible to the user. Four bytes are used to represent each color voxel, the first represents the blue value, and the second and third represent the green and red respectively. The fourth byte is reserved for the alpha channel and in general should be left filled with zero. For the example plug-in we assign the voxel intensity of the first source to the red channel, the voxel intensity of the second source to the blue channel, and set the green channel to zero.

The final function of the class is not required, but can be used to place appropriate text in the title bar of windows that are using the plug-in. This function is named Initialize and can be found on line 30.

After a successful build, the plug-in assembly (DLL) must be placed in the 'plugins' folder found in the Fusion Viewer application directory. After restarting the Fusion Viewer application the new plug-in will show up as an option under the 'Fusion' menu. The option will share the same name as the assembly containing the plug-in.

A screenshot taken after running Fusion Viewer and providing the sample plug-in a MRI and PET image can be seen in Figure 8.

4. DISSCUSSION AND CONCLUSION

Fusion Viewer, a new application for visualizing and examining three-dimensional medical data sets, was presented. The most significant advantage of Fusion Viewer over previous applications is its capabilities in examining multiple data sets simultaneously. Not only does Fusion Viewer keep multiple data sets synchronized, but it also provides tools for displaying them as a single color data set. Several fusion options are already provided as plug-ins, and a simple interface exists for experimenting with new fusion schemes.

In addition to these capabilities Fusion Viewer provides interfaces to evaluate various projection techniques, such as the popular MIP. Specifically of interest are its capabilities to fuse projected data sets, and create projections of fused data sets. Fusion Viewer's capability for dealing with the dynamic range of medical images was presented. Novel spline based techniques introduced in the Fusion Viewer software package were presented, and an interface exists for experimenting with new dynamic range techniques.

Fusion Viewer was implemented on the .NET framework for easy distribution, installation, and compatibility with current and future platforms. Its design for only processing the data currently being presented to the user makes it an ideal tool for real-time fusion and evaluation of large data sets.

The latest version of Fusion Viewer can be found at [12]. The most recent news, the status of known bugs, and instructions for submitting feature requests and bug reports are provided there as well. The software and support are currently provided at no cost.

REFERENCES

1. Microsoft Corporation, "Technology Overview," <http://msdn2.microsoft.com/en-us/netframework/aa497336.aspx>.
2. H. Drolon, "FreeImage", <http://freeimage.sourceforge.net/>.
3. H. Drolon, *FreeImage Documentation, Library Version 3.9.2*, Oct. 29, 2006.
4. F. S. Hill, Jr., *Computer Graphics Using Open GL*, Second Edition, Prentice Hall, Upper Saddle River, NJ, 2001.
5. J. Beutel, H. L. Kundel, R. L. Van Metter, eds, *Handbook of Medical Image Volume 1: Physics and Psychophysics*, SPIE Press, Bellingham, Washington, 2000.
6. R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, Second Edition, Prentice Hall, Upper Saddle River, NY, 2002.
7. "ImageJ," <http://rsb.info.nih.gov/ij/>.
8. K. G. Baum, M. Helguera, A. Krol, et al, "Techniques for Fusion of Multimodal Images: Application to Breast Imaging," *Proceedings of the 2006 IEEE International Conference on Image Processing (ICIP)*, Atlanta, GA, Oct. 8-11, 2006.
9. K. G. Baum, M. Helguera, A. Krol, "Genetic Algorithm Automated Generation of Multivariate Color Tables for Visualization of Multimodal Medical Data Sets," *Proceedings of IS&T/SID's Fourteenth Color Imaging Conference (CIC)*, Scottsdale, AZ, Nov. 6-10, 2006.
10. K. G. Baum, "Publications," http://www.kgbtechnologies.com/karlbaum/resume_publications.html.
11. J. Sharp, J. Jagger, *Microsoft Visual C# .NET Step by Step*, Microsoft Press, Redmond, WA, 2002.
12. K. G. Baum, "Fusion Viewer", <http://www.kgbtechnologies.com/fusionviewer/>.